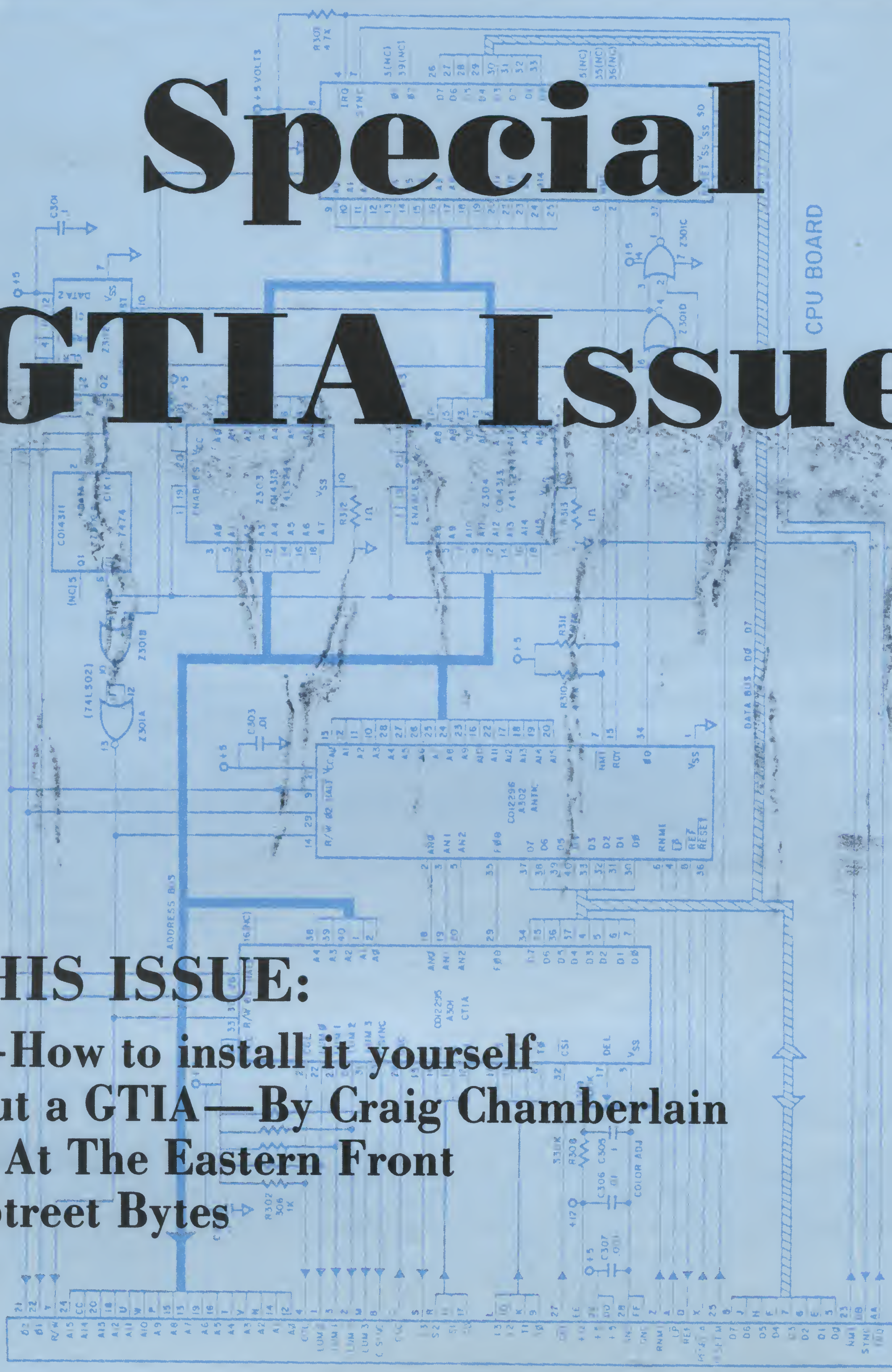
March, 1982
NEWSLETTER

$2.00

Michigan Atari Computer Enthusiasts



Special GTIA Issue

IN THIS ISSUE:

GTIA—How to install it yourself
All about a GTIA—By Craig Chamberlain
Victory At The Eastern Front
Baker Street Bytes

READY FOR A REAL CHALLENGE?

It's a battle of wits and fast action, this challenging game in ROM form for *Atari® 400™ or Atari® 800™ Personal Computer Systems. You're trapped in an Alien Fortress. Your mission is to clear your sector of the Alien Droids and progress to the next sector. The first sector seems easy enough . . . BUT–WATCH-OUT . . . as you progress from sector to sector, the Droids become more numerous and aggressive. A great challenge and fun too!

K-RAZY SHOOT-OUT is destined to be a favorite of Atari enthusiasts across the country with features like:

- Solid-state ROM pack
- Seven progressive levels of play
- Every game different
- Bonuses and penalties
- Full-color T.V. graphics and space-age sound
- Full-color instruction booklet
- Millions of variations
- A constant challenge

[WARNING: Playing the game could be HABIT forming]

Now at your Atari Dealer!

Coming soon:

- K-DOS™
- K-STAR PATROL™
- K-RAZY KRITTERS™

*Atari is the Registered Trademark of Atari, Inc.

FOLD & TEAR

If not available in your area, send a check or money order directly to:

K-BYTE • P.O. Box 456 • 1705 Austin Street • Troy, Michigan 48099

Toll Free Phone Orders: **1-800-521-0560** • In Michigan: **(313) 524-9878**

☐ $49.95, plus $1.50 shipping and handling, Michigan residents add 4% sales tax.

☐ $1.00 for instruction book only

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ________ ZIP ________

Please allow 3 weeks for delivery. ☐ **MasterCard** ☐ **VISA**

CHARGE CARD NO. ____________________

EXPIRATION DATE ____________________

FEBRUARY MEETING MINUTES

Sheldon Leemon, Secretary

FEB CONFAB SRO AGAIN. Since it was the first time for about 60 of the 250 people in attendance, we were gentle. The Show-and-Tell portion of the program was devoted to the GTIA. Which, as Craig Chamberlain explained, is not Arlan's new sports car, but a large-scale integrated circuit that has superceded the older CTIA, the television interface chip that was included with computers built before 1982. This upstart silicon overachiever supports graphics modes 9, 10, and 11 (Huh?). Those in need of further illumination will be pleased to read the GTIA-related material elsewhere in the newsletter. Without any long-winded explanations (he saved those for his article), Craig demonstrated some of the kinds of graphics effects that these new modes afford. Of particular interest to those in attendance was the demonstration of the excellent modelling of, ahem, forms available from graphics mode 9, with its 16 gray-scale tones.

During the business portion of the meeting, the membership approved the distribution of newsletters at the meetings, with no-shows receiving their issues from Uncle Turtle (who we use when it absolutely, positively, has to get there eventually). The brethren (and sistren) also authorized the expenditure of $300 for a partially-subsidized 800 from Family Computer Center, so that the MACE bulletin board can serve Atari owners everywhere from noon to noon daily. Along with donations of a Hayes Smartmodem from Rite-Way, a 32K memory board from Binary, and the loan of 4 disk drives, 822 printer, and an 850 from a friend, our BBS is fully factory equipped. This eighth wonder of Southeastern Michigan is accessible to the modem crowd at (313) 868-2064. And of course, all praise to our own Terminal Man, software author Tom Giese, to whom all praise is due.

Reports from Special Interest Group interim leaders revealed that most groups were snowed out of a first meeting. When more clement weather conditions allow for more permanant arrangements, you will read it here first (if you don't see it on the BBS before).

The main program was a slide show entitled "Data Bases of Many Lands", and provided enlightenment and amusement to those not stupified by the length of the proceedings theretofore. Many thanks to our guest speaker, whose identity will go unrecorded, due to my pathetically ineffecient method of storing important information. I don't think that even a computer data base could help; I'd probably lose the computer.

THE FAMILY COMPUTER CENTER

THE GTIA IS HERE

by Craig Chamberlain 3/6/82

The GTIA is an exciting new graphics chip now being shipped in ATARI 400/800 computers. Among its special features are a sixteen color mode with a resolution eight times better than the APPLE's, and the capability of generating two hundred fifty six color variations. The GTIA chip provides three new graphics modes in addition to the normal fourteen totally different full screen modes. This article defines a few terms relating to graphics, explains the normal graphics modes, then introduces the new modes provided by the GTIA.

We all know that the ATARI 400 and 800 have superior graphics capabilities. This has been achieved by designing special chips to handle video display tasks, taking that burden off the main microprocessor. In the ATARI computers these special chips are known as ANTIC and CTIA.

The ANTIC chip is actually an advanced DMA (direct memory access) controller that qualifies as a true microprocessor. It has an instruction set (mode lines and "load memory scan" operations), a program (the good 'ole display list), and data (display memory and character sets). Chris Crawford made this point well in his September 1981 BYTE article. The only thing he did not tell us was how the wizards at ATARI came up with the name ANTIC.

This special chip is a rather busy fellow. Its responsibilities include doing DMA for the display list, the display data (playfields), the character set, and player/missile buffers. Besides that it sets the playfield width, controls horizontal and vertical fine scrolling, keeps track of the vert=ical position of the scan beam, and handles NMI interrupts. It also supports a light pen.

So much for ANTIC. The other chip is the CTIA, or Computer Television Interface Adapted integrated circuit. This is the chip which handles all color and luminance (brightness) information to send to the television screen. This is a complicated process, but the chip designers at ATARI got carried away and created whole new functions which we know as the player/missile graphics system. It is the CTIA that processes the horizontal position, size, priority, and color of the players. The CTIA also watches for player/playfield collisions, joystick triggers, and console keys, so like the ANTIC, it is a busy chip.

The new GTIA chip replaces the CTIA. Rumor has it that the "G" stands for George. Apparently some fellow named George was still not satisfied with all the special functions of the CTIA, and gave it the ability to generate three totally new graphics modes. When you find out what these new modes can do, I think you will really like George and his GTIA.

(Actually, the truth of the matter is that the GTIA was originally intended to be included in the ATARI computers when they were first released. When the chip was not ready at production time, the CTIA, a

continued

GTIA (continued)

compromise chip, was used.)

The three new modes are 9, 10 and 11. The operating system and, therefore, ATARI BASIC, supports these new modes. But before I describe all the features of these new modes, I want to define a few essential terms and review the normal graphics modes 0 through 8.

In order to fully understand ATARI graphics, one must have a solid concept of how a television display is generated. And no discussion on "television theory" would be complete without a definition of the "color clock". The term color clock arises from the fact that there is a problem in measuring distances on a television screen. Different television sets have different screen sizes, with 9", 13" and 19" being common diagonal measurements. All television sets, however, have a scanning beam which translates a signal from the computer into a picture on the screen.

The signal coming from the computer contains two characteristics. It has a frequency, which defines a color, and it has an amplitude, which defines the luminance of that color, often referred to as the brightness or intensity. These qualities of the computer signal affect the way in which the scanning beam shoots electrons at the phosphors on a television screen. This electron shooting process is done horizontally one line at a time, but it is done so quickly as to not be noticeable to the human eye.

In drawing a line the scanning beam starts at the left edge of the screen and proceeds to the right edge, shooting electrons the whole time. Since the beam has a finite amount of time it can spend drawing one line, the beam will seemingly have to move faster to cover more area on a larger screen. Thus the problem of trying to measure horizontal distances is further complicated by the fact that different scanning beams not only travel different areas but also at different rates. Our unit of measurement cannot really be a distance; it must be a unit of time. The hint I gave a moment ago was that the scanning beam has a certain amount of time it can spend on one scan line. How fast or how far the beam travels is insignificant.

The fact that our unit of measurement is based on time explains the word "clock" in the term "color clock". Now let me present my personal definition for this term.

A color clock is the amount of time the computer needs in order to sufficiently change the frequency of the signal it generates so as to produce a different color.

Wow! What a mouthful! Well, read it a couple of times and see if you can figure it out. Remember that it's my own personal definition; it has worked for me, but some people may not agree with it. Here's another definition.

A scan line is the horizontal path of the scanning beam from the left edge of the screen to the right edge.

Scan lines extend horizontally across the screen, but it takes a lot of them stacked vertically to fill up the screen from top to bottom.

continued

GTIA (continued)

Therefore, horizontal resolution is usually expressed in terms of color clocks while vertical resolution is expressed in scan lines. On different television sets the actual lengths will of course differ, but the resolution horizontally to vertically is always proportionate. It turns out that on any screen, one color clock appears to be equal in length to two scan lines.

Now we get technical. The scanning beam starts at the upper left corner of the screen and travels horizontally to the right. By the time it hits the right edge it has drawn one scan line that is 228 color clocks wide. The beam then shuts off for a short period while it returns to the left edge, only one scan line lower. This period is called the horizontal blank because the beam is turned off and it is returning to the left edge. The beam then turns on again and starts drawing the next scan line. This sequence of drawing scan lines continues for 262 times. At that point the scanning beam, at the lower right corner of the screen, shuts off and returns to the upper left corner of the screen during a period known as the (guess what!) "vertical blank". The whole process of drawing 262 scan lines, each of 228 color clocks, plus the blanking periods, constitutes one "frame". The television draws sixty of these frames every second, because your home power line is 60 Hz (cycles). The name given to this display method is "raster scan". The fact that your ATARI follows a broadcast standard referred to as "NTSC" makes it one of the few home computers that can be video taped without special equipment.

Just because the scanning beam generates all those scan lines and color clocks doesn't mean that the computer is generating that much display data. Even if the computer did, you wouldn't see the whole image since most television sets display a little less than two hundred scan lines of about one hundred seventy color clocks. The part where the true picture exists is called the playfield, and it's time for another definition.

The playfield is the portion of each scan line for which data read from memory can produce colors and luminances. The background exists at the ends of each scan line; the playfield is in the middle. From the viewpoint of one frame, the playfield appears as a rectangular region which extends to the sides of the screen.

Two things control the size of this playfield area. The heighth in scan lines is controlled by the display list as you will see in a moment. The width in color clocks is set by the DMA control register of the ANTIC, as mentioned earlier.

```
SDMCTL  $022F    559  (shadow)
DMACTL  $D400  54272  (hardware)

D5       1  display list DMA enable
         0  display list DMA disable

D1,D0   00  playfield DMA disable (no playfield)
        01  narrow playfield (128 color clocks)
        10  standard playfield (160 color clocks)
        11  wide playfield (192 color clocks)
```

CONTINUED ON PAGE 28

BAKER STREET BYTES

March, 1982

By RICHARD GIZYNSKI

I'd like to start this month's column out with a thank you to Bill Wilkenson for his helpful and constructive criticism. I can use a lot more of that kind of 'picking on.' I have gained a lot of insight on how Atari Basic works from Bill's articles in the last few issues of Compute. Maybe Bill would like to give us a freebie and show us some of his Basic A+.

One more thing before I begin this months program. When I started programming, I used short and cryptic variables to save space in memory. Then I learned that the name of a variable is kept in a special table. In the rest of the program, the variable is represented by a one byte number called a token. The only thing I saved by using short, cryptic variables was the trouble of typing in a longer name and a couple of bytes in the name table. I lost the ease with which I could trace the flow of my programs.

Last month, I advocated the use of DATA or Rem statements to hold the letters and articles you wanted to pass on to your friendly editor. Here is a easy way to use them. You also get a handy routine that can be used to draw and save game boards etc. With this program you can write and edit any length article one screefull at a time. You can use all of the deleting and inserting features of the screen editor. Then you save that screenfull and go on to the next.

Lines 10-220 initalize the program. Lines 110-130 set up string lengths based current length of your screen line. The only character space that you can't use is the very last one on the screen. Printing to that space would cause the screen to scroll and you would lose the line at the top. Lines 140 to 220 allow the program to check and see if there are already stored DATA statements stored in the program. This prevents you from overwriting something that you had already saved. The TRAP on line 150 directs the program to a GOTO menu line when you run out of read statements. I should have TRAPed directly to the menu, but I thought it would make this section seem little more complete understand. The RESTORE 10000 on line 160 casues the program to skip over the DATA statements that hold the instructions.

Lines 230-280 GET a key from the keyboard and and translate it to a string character. I used this so commands could be used that would not print on the screen. 260 adds the characters to a string called WORD$. WORD$ keeps you from losing character groups that started like a command. When you press the ATARI (reverse character) key and follow it with the less-than key and the capital letters STOP, the program will switch to the saving routine. Line 270 protects WORD$ from getting longer than it DIMensioned length of 6.

Lines 290 to 450 decode the command to stop printing to the screen and start saving the screen. Any combination but the completed command will simply print to the screen.

When the command to stop is read, lines 460 to 580 change the color of the screen (so that you know the program is working) and GET the characters from the screen and place them in a string called SCREENFULL$. The string is used as temporary storage untill the screen can be cleared and the information put into DATA statements. The commas have been reversed because a comma is used in a DATA statement to mark the end of a DATA set and will not be read as information.

DATA statements are used to store the screen information because they can be LISTed and then modified easily. You can also LIST "C:",10000,30000 to save the lines for use elsewhere. Merging them with another program is done by ENTER "C:" after the other program is loaded.

Lines 590-730 do the trick of storing SCREENFULL$ in DATA statements. An asterisk is put in at the beginning and end of each line both to mark their beginning and ending and because Atari will ignore the spaces on the front end of a DATA statment. When you type in line 610 put REM in front of the SETCOLOR statement until the program is debugged. The screen is cleared, the cursor POSITIONed near the top of the screen and a line number and DATA are printed. This is followed by the segment of SCREENFULL$ that corresponds to each screenline. The word 'CONT' is PRINTed a seperate line and will be interpreted as the command to CONTinue.

After the cursor is rePOSITIONed, line 710 POKEs a read from the screen command that is picked up by the screen editor after the STOP command is executed. Strange, but it works. This allows the screen editor to tokenize the DATA statement for you instead of having to write a routine that finds the end of the program and tokenizes the line number and the DATA statement etc. Line 720 is activated when the screen editor reads CONT from the screen. It is the command to write to the screen and it keeps the editor from crashing the program. Line 730 insures that all of the screen lines have been copied. Lines 740-800 gives you the option of continuing to write to the screen. I added the POKE 694,0 (turn off reverse character) because I kept forgetting to hit the Atari button after my stop command.

Lines 810-980 let you read what you have saved, adding one line at a time to the screen. That allows a transition between different screenfulls. If this seems to slow just bypass this section by adding a line:

```
845 GOTO 1260
```

When you want the next screenfull, press any key. If you don't have a printer, you can ignore lines 980 to 1080 which read and LPRINT the DATA statements.

Lines 1090 to 1210 contain the menu. I like the feature of GETing the key and not having to press return. GET returns a number value for the key. This is translated by line 1190 into something that can be used by the ON-GOTO in line 1210. Line 1200 checks to make sure your choice is a valid one.

The instructions could be left out but they are an easy way to test the program. After entering lines 1220-1320, RUN the program to initalize the strings. When the menu appears, press the BREAK key. Now type:

```
LINE# 1330
GOTO 1100
```

The menu will appear on the screen. Now, type in the first group of lines. They will be converted to DATA statements to be retained in the program. You don't have to type

continued

anything for the last blank line. You do have to leave these blank DATA staments in the program. After you type the 'PRESS ANY KEY TO CONTINUE' line, press the Atari button and type STOP. The screen will turn pink to let you know that it is working. After about a minute, the screen will turn blue. If you put the REM in the SETCOLOR command that I mentioned above, you will see the DATA statements being formed on the screen.

TO USE THE SCREENWRITING AND SAVING PORTION OF THE PROGRAM, JUST ENTER ONE ON THE MENU. BE CAREFULL NOT TO INSERT A LINE THAT WOULD CAUSE THE SCREEN TO SCROLL OR YOU WILL LOSE THE LINE THAT SCROLLS.

YOU MAY USE ANY NORMAL, GRAPHICS OR CONTROL KEY WHILE YOU ARE "ON SCREEN" COMMAS, HOWEVER, ARE SAVE IN THEIR REVERSE FORM.

WHEN YOU WANT TO END WRITING TO THE SCREEN, JUST PRESS THE ATARI SYMBOL AND TYPE <STOP. WHILE YOU ARE TYPING THIS COMMAND, THE COMMAND WILL NOT APP EAR ON THE SCREEN UNLESS YOU MAKE A MISTAKE IN TYPING.

THE SCREEN WILL THEN TURN PINK TO INDICATE THAT THE SCREEN INFORMATION IS BEING PUT INTO A STRING. WHEN THE SCREEN TURNS BLUE, THE INFORMATION IS BEING PUT INTO DATA STATEMENTS.

PRESS ANY KEY TO CONTINUE

WHEN YOU HAVE FINISHED SAVING DATA FROM THE SCREEN, YOU MAY REVIEW WHAT YOU HAVE SAVED. YOU MUST PRESS A KEY FOR THE NEXT LINE TO APPEAR ON THE SCREEN. I ADDED THIS FEATURE SO THAT YOU MIGHT SAVE SEVERAL SCREENFULLS AND COULD LOOK OVER EACH AT A SLOW PACE.

IF YOU ARE USING THIS PROGRAM TO MAKE A GAMEBOARD, REMEMBER THAT THE LAST CHARACTER SPACE ON THE SCREEN CANNOT BE USED OR THE SCREEN WILL SCROLL FOR THE NEXT LINE.

I PUT AN ASTERISK AT THE BEGINNING A ND END OF EACH LINE TO MAKE AFTER-THOUGHT EDITING EASIER AND TO SAVE THE SPACES AT THE FRONT END OF A LINE.
IF YOU NEED TO ADD EDIT LINES, JUST TY PE OVER THE CHARACTERS IN THE DATA STATEMENTS. TO ADD LINES, JUST ADD DATA STATEMENTS.

PRESS ANY KEY TO CONTINUE

WHEN YOU HAVE COMPLETED YOUR WRITING JUST LIST LINES 10000 TO 30000. THEY MAY THEN BE ADDED TO ANY GAME OR OTHER PROGRAM.

IF YOU ARE WRITING TO MACE, CSAVE THE WHOLE PROGRAM. IT'S QUICKER THAT WAY.

```
10 REM POOR MAN'S WORD PROCESSOR AND
20 REM SAVE SCREEN UTILITY
30 REM BY RICHARD GIZYNSKI
40 REM  SET UP PARAMETERS
50 LM=PEEK(82)
60 RM=PEEK(83)
70 WIDTH=RM-LM+1
80 AREA=WIDTH*24
90 DIM SCREENFULL$(AREA)
100 DIM SCREENLINE$(42)
110 DIM CHARACTER$(1),WORD$(6)
120 OPEN #2,4,0,"K:"
130 OPEN #3,4,0,"S:"
140 REM  CHECKS PROGRAM OLD DATA
150 TRAP 210
160 RESTORE 10000
170 LINE=10000
180 READ SCREENLINE$
190 LINE=LINE+10
200 GOTO 180
210 TRAP 40000
220 GOTO 1100
230 REM  GET KEY INPUT
240 GET #2,KEY
250 CHARACTER$=CHR$(KEY)
260 WORD$(LEN(WORD$)+1)=CHARACTER$
270 IF LEN(WORD$)=6 THEN WORD$=""
280 RETURN
290 REM  CHECK FOR STOP COMMAND
300 GRAPHICS 0
310 GOSUB 240
320 IF CHARACTER$="<" THEN 350
330 PRINT CHARACTER$;
340 GOTO 310
350 WORD$=CHARACTER$
360 GOSUB 240
370 IF CHARACTER$<>"S" THEN 440
380 GOSUB 240
390 IF CHARACTER$<>"T" THEN 440
400 GOSUB 240
410 IF CHARACTER$<>"O" THEN 440
420 GOSUB 240
430 IF CHARACTER$="P" THEN 470
440 PRINT WORD$
450 GOTO 310
```

continued

```
460 REM  STORE SCREEN IN STRING
470 SETCOLOR 2,4,4
480 POSITION 2,0
490 L=0
500 FOR VERT=0 TO 23
510 FOR HORZ=PEEK(82) TO 39
520 POSITION HORZ,VERT
530 L=L+1
540 GET #3,KEY
550 IF KEY=44 THEN KEY=172
560 SCREENFULL$(L,L)=CHR$(KEY)
570 NEXT HORZ
580 NEXT VERT
590 REM  STORING SCREEN
600 REM IN DATA STATEMENTS
REM 610 GRAPHICS 0:SETCOLOR 1,9,4
620 ITEM=0
? — 630 PRINT "}" ESC CONTR CPL
640 POSITION 2,2
650 L=ITEM*WIDTH+1
660 PRINT LINE;" DATA *";SCREENFULL$(L,L
+WIDTH-1);"*"
670 ITEM=ITEM+1
680 LINE=LINE+10
690 PRINT "CONT"
700 POSITION 2,0
710 POKE 842,13:STOP
720 POKE 842,12
730 IF ITEM<24 THEN GOTO 640
740 REM  CONTINUE?
750 GRAPHICS 0:? "WOULD YOU LIKE TO CONT
INUE"
760 PRINT "WRITING TO THE SCREEN"
770 POKE 694,0
780 GOSUB 240
790 IF CHARACTER$="Y" THEN 300
800 GOTO 1090
810 REM  PRINT OUT SCREEN DATA
820 RESTORE 10000
830 GRAPHICS 0
840 TRAP 930
850 PRINT "PRESS ANY KEY EXCEPT M FOR EA
CH LINE"
860 PRINT "PRESS M FOR MENU"
870 PRINT
880 GOSUB 240
890 IF CHARACTER$="M" THEN 1100
900 READ SCREENLINE$
910 PRINT SCREENLINE$(2,WIDTH+1);
920 GOTO 880
930 PRINT "THAT'S ALL FOLKS."
940 PRINT "PRESS ANY KEY FOR MENU"
950 GOSUB 240
960 TRAP 40000
970 GOTO 1100
980 REM  PRINT OUT
990 GRAPHICS 0
1000 POSITION 2,6
1010 PRINT "TURN ON PRINTER AND HIT ANY
KEY"
1020 GET #2,KEY
1030 RESTORE 10000
1040 TRAP 930
1050 READ SCREENLINE$
1060 SCREENLINE$=SCREENLINE$(2,WIDTH+1)
1070 LPRINT SCREENLINE$
1080 GOTO 1050
1090 REM MENU
1100 GRAPHICS 0
1110 POSITION 17,6
1120 PRINT "MENU"
1130 PRINT :PRINT "1. ENTER SCREEN DATA"

1140 PRINT "2. REVIEW PREVIOUS DATA"
1150 PRINT "3. PRINT SAVED SCREEN TO PRI
NTER"
1160 PRINT "4. INSTRUCTIONS"
1170 PRINT :PRINT "YOUR CHOICE"
1180 GET #2,KEY
1190 KEY=KEY-48
1200 IF KEY<1 OR KEY>4 THEN 1170
1210 ON KEY GOTO 300,820,990,1220
1220 REM  INSTRUCTIONS
1230 GRAPHICS 0
1240 RESTORE
1250 TRAP 930
1260 FOR I=1 TO 24
1270 READ SCREENLINE$
1280 IF I=24 THEN PRINT SCREENLINE$(2,WI
DTH);:GOTO 1300
1290 PRINT SCREENLINE$(2,WIDTH+1);
1300 NEXT I
1310 GET #2,KEY
1320 PRINT
1330 GOTO 1260
```

Ⓜ

```
******************************
* The day, third Thursday,  *
* The group is M.A.C.E.,    *
* The time, 7:30,           *
* Berkley's the place!      *
*                           *
* BURMA SHAVE               *
******************************
```

THE GAMES WE PLAY

VICTORY ON THE EASTERN FRONT

By RICHARD GIZYNSKI

Eastern Front has a combination of tactical and strategic gaming ability. In the tactical sense, the artificial intelligence created by Chris Crawford is excellent. The Russians never have problems in the traffic jams that frustrates the German player. If the strategic ability of the Russians were as great as their tactical ability it would take another computer to achieve the 255 score that is possible.

To get that grand score of 255, I had to relate to the historical situation that the game represents. I looked at the game from two viewpoints, the tactical and the strategic. The orders that are given, are simply that, orders. They are executed only if they are 'legal' at the time of execution. That's the tactical part of the game. Issuing orders that can be carried out. The strategic parts of the game, are the allocation of armies to a given area, varying the tactics used during the different seasons, coping with the large numbers of Russian troops. This is where you can beat the tactically oriented intelligence.

Don't worry about giving the computer extra time to make its moves. The artificial intelligence acts mostly in response to your moves. If you haven't done anything, The Russian side will not be able to calculate what it should do. Also, its refinement of its move is not as great as you might think. It will cost you far more for careless moves than you will lose by taking more time to move.

When you give orders for the each move, remember that the orders will be carried out only if they are 'legal' at the time of execution. If they are 'illegal', the unit will simple wait for them to become legal. You have to change the orders or the piece will stay put till it is destroyed. You must give orders that will cover a situation where rapid advances are made. Give orders to each unit based on what you hope it will do for more than one turn, but be ready to change those orders as the situation changes. I issue orders to each piece on every turn.

One of the main tactics of the Russian side is to create impasses and road blocks with a checkerboard pattern of unit deployment. This leaves the German player attacking an empty square and unable to move due to 'zones of control'. The second basic tactic is the placement of two pieces in a line adjacent to the German 'line of advance'. This keeps the German from continuing his advance due to the rule that you can't leave one full square zone of control and go into another unless it is the result of an attack on a piece that was there. Since these are the main tactics of the Russian during the first part of the game, they are difficult to anticipate and you must issue orders to back a piece out of a zone and then attack another square. I would also attack the blocking column with another, parallel piece. This forces the Russian column to crumble and allows your locked piece to complete its maneuver.

Traffic jams, due to successfully attacking a piece with two or more units, can be avoided by giving orders to back out of the square after you advance into it. You can give orders to go on to attack another square after you are finished with the first square. The orders can be cancelled if the 'enemy' has a retreat, but this will increase the effectiveness of your attack.

Now for some stratagy. During the actual attack on Russia, Hitler simply wanted to take lots of territory. He sent his troops into Russia in three main groups with ill-defined objectives and simply tried to 'grab off' a lot of territory. He had a stated policy of not giving up any of the ground that he had taken. He also considered the German soldier 'superhuman', and felt that understrength units should continue to press the attack. If you follow this plan you will have the same results that Hitler had.

A much wiser course was suggested by Hitler's general staff. First, a general all-out push to punch some holes in the Russian line. Next, begin a process of destroying trapping and destroying enemy units. Your army is much more effective if you have 'cut off' a Russian unit. Third, set LIMITED objectives in the advance. With the main objectives of a drive being the destruction of the Russian army, and the taking of Moscow. When Moscow is held, the Russians gradually become less aggressive and their replacement rate slows. Be willing to 'attack to the rear' or retreat if pieces are trapped or greatly understrength. Leave minimum forces to deal with Russians in your rear area. One or two Russians are not a major cause for concern, but several will block your own supply lines and make your fighting forces anemic. Last, try always to fight at a 'local advantage'. That is, get more troops in a sector of the board and use that advantage to surround and destroy.

The first set of orders that I gave, were for both the first and second row of troops to advance forward eight moves. On the two 'corners', I had both flank units attack the corner first, then withdraw and attack in a frontal direction. The northermost troops headed north, the main line headed east, and a few contingents in the south headed east, two then North. I moved the Finish troops up to Leningrad. At each turn, I modified the orders to correspond with what I saw. Some Panzer units had passed a few trapped Russian units. These Russians were kept surrounded and attacked by infantry following the tanks. In the east, the main Russian line had fallen back several 'squares'. I pursued them to the edge of the marshes with the forces that had broken through. I usually gave my units more orders than they could carry out. If they finished what they were doing in combat, they would move up to the front and a fresh attack.

Before my forces hit the Prippet Marshes, I sent most of the armour and about half of the infantry north. I wanted the tanks to stay out of the marshes and keep on faster ground. I sent them along slightly different routes to avoid traffic jams. Two panzer units headed east then south to trap the Russian forces in the south. Only one armour unit was committed to the marshes to destroy the Russians that the infantry couldn't reach in time. I had only a little more than a checkerboard pattern of infantry attacking the troops in the marshes.

The limited objectives were to surround SMALL pock-

continued

ets of Russian units and destroy them and to open 'corridors to the north and south where panzers could operate more effectively. At this stage of the game, if you try to surround a large pocket of Russians, they merely 'strategically withdraw'. The panzers headed toward the north also had the objective of getting past the Prippet Marshes on a narrow front. Half of the north bound infantry were sent with the panzers the other half were left to 'mop up' Russians that headed north and west. The southern Russians were a little easier to surround. They tried to slow the northward advance from the southern-most Germans as well as block the eastward advance from the main line.

By the end of August, my northern forces had advanced far enough to the east to split off two panzer and two infantry units that were given the objective of taking Leningrad. My southern forces had trapped most of the Russians that had been south of the Prippets and were nearing the gateway to the southern reaches of the central plains. The southern forces were very scattered. The units in the Prippets were still advancing and surrounding some of the Russians there. Most of the southern German units were small and I used the small units to 'trap' Russians. The southern panzers were forcing a corridor to the east with only two infantry to help keep them from getting trapped themselves.

I kept careful watch on units that had gotten trapped then freed. I avoided using them in the front line until they had rebuilt most of their strength. If possible, I gave each major piece a 'rest' in September to renew their strength.

On September 14, Moscow fell to my northern panzer armies. But I still had a large force of Russians to mop up or push east. My southern armies were scattered out but heading east and the troops in the marshes were now out. I used the last two weeks of September to clear a little space around Moscow and to move my panzers to the rear for 'rest and recreation'. Leningrad fell on September 28 and there I moved my infantry as far east as I could. I advanced the panzers a few squares to the east and move the Finns in to occupy Leningrad.

During the month of October, my army stood on the defensive. I had only two mop-up operations in the south central plains. The rest of my troops avoided combat as much as possible. The only exception to this was not giving ground near Moscow. Generally, I just pushed the START button to get rid of the month with as few operations as possible.

In November, I started a push toward the arctic circle. Several infantry and two panzers moved toward Gorky. This was to swing around the northern reaches of the Russian line. The Russians made their 'Winter Offensive' but the holding of Moscow took much of the fight out of them. I kept up the tactic of surround a few and attack them. I opened up a large part of the northern front.

Early in January, the score had reached the fabled 255 mark. I had to continue to rely on the fact that the Russians lost much of their fighting strength when cut off to maintain that score. Almost all of my replacements had been sent to the north to reinforce the armies fighting there. The German losses were high, but the Russians were higher. When a German unit got depleted, I pulled it out of the fight for a few turns if I could.

In February, the surviving Russians were all behind the Volga River and in the Crimia. I was unable to completely destroy all of the Russians but that is not necessary to get and keep the 255 score. Ⓜ

HARDWARE CORNER

THE ATARI 800 GTIA CHIP FIX

Tired of having only 9 graphics and text modes? You may now have **12** with the **GTIA** chip fix. You presently have a graphics chip called the CTIA. We will now go through the procedures to swap these chips.

You will need the following items,

1. I.C. number CO14805—01 (GTIA)
2. Medium size Phillips head screwdriver (non-magnetic)
3. Small size regular head screwdriver (non-magnetic)
4. A work area which is not vulnerable to static electricity.
5. About 10 minutes of uninterrupted time.

1. Open the top of the 800. Remove any cartridge which may be inserted. Now, open and remove the entire top door assembly and set it aside. Remove all RAM and ROM boards.

2. Turn the 800 over with the front controller ports facing you. Using the phillips head screwdriver, remove the three screws along the front area of the bottom cover. Now, remove the two screws that are located towards the back of the bottom cover.

3. Gently lift off and set aside the bottom cover.

4. In the center of the computer you should see a rectangular metal plate held in place by eight screws. Remove these screws and set them aside.

5. Now, down even with the circuit board are three more screws you must remove. The first and second are located along the left and right edges of the seamless metal case. The third is over to the far left of the circuit board, about two inches from the front.

6. Here come the tricky parts. You may want to get someone to assist you. You will find that the plate assembly is connected to the lower circuit board at the left-front corner. This is a pin connector so you must be careful when pulling the boards apart. Grasp the plate assembly near the connector and lift gently but firmly STRAIGHT up. Only lift it up until the pins have cleared the holes.

7. Let the plate assembly rest on the top of the circuit board. Reach around to the rear of the board and gently lift up and towards you. If you look under the board you will see that it is connected to the rest of the computer by two connectors. One is at the front and must not be twisted too much. This is where it may be nice to have someone helping you.

8. Slide the plate assemble part-way out of the metal case. In the position that would equate to behind the RAM/ROM slots is a circuit board. This is where the CTIA chip is.

9. This smaller circuit board should be marked "CPU". With firm but gentle pressure, pull this board out of its connector. Remember. When you put it back in the chip side of the board must be facing away from the RAM/ROM board slots.

10. Using the small screwdriver (or better yet, a special chip removal tool) verrrrrrry carefully remove the CTIA chip (see illustration). You may have to pry a little on one end and then a little on the other to get it loose. Gently place the chip where it will not get damaged. REMEMBER, the chip is still good.

11. With the old chip out you will now place the new GTIA chip in. First, look at the socket on the board. See the indentation in one end of it? Or possibly a dot? There is a corresponding indentation on the chip. Line these up. THEY MUST BOTH BE AT THE SAME END!

12. Carefully insert the new chip into the socket. Before you push it all the way in, check to see that all of the pins are going into the socket and that none of them are bent. If all is well, push the chip all the way into its socket.

13. Reassemble the computer by reversing the previous instructions. Be careful with the connections. If you should accidently pull one of them apart, just reconnect it. BUT . . . make sure that it is connected properly.

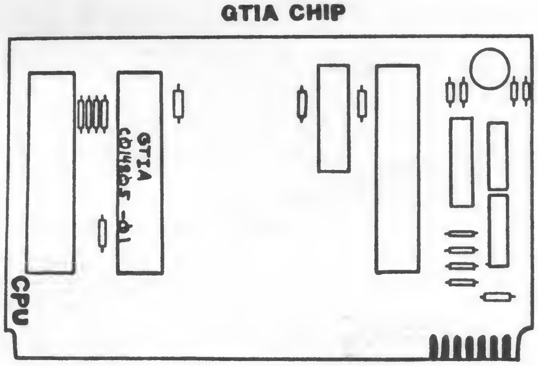
The GTIA supports the nine graphic modes of the CTIA plus three additional modes. Mode 9 supports BG plus FG of 16 different luminences with the color of the BG. Mode 10 supports BG and eight playfields, each of which may be set to 16 different colors and eight luminences. Mode 11 supports BG plus FG of 16 different colors, using the luminence of the BG. PLOT, DRAWTO, and the FILL procedure work the same as in Modes 3-8. Mode 9 allows the user to plot points in one of 16 different luminences which use the color of the background (register 4). Points in up to 16 luminences are displayed simultaneously. The background color/luminences are displayed simultaneously. The background color/luminence is set with: SETCOLOR 4, color, luminence. The luminence of the next foreground point to PLOT is set with: COLOR 0-15.

MODE 10 allows the user to plot BG and points of eight different color/luminence values. The different playfields may be selected by the command: COLOR(0-7). The command, SETCOLOR (0-4) color, luminence sets the color/luminence; 0-3 for the playfields and 4 for the BG. The command: POKE (704-707), color luminence value (0-255) sets the color/luminence for the playfields 0-3. The color/luminence of the playfield 4-7 can also be set using POKE (708-712) instead of using SETCOLOR.

MODE 11 allows the user to plot points in one of 16 different colors which use the luminence of the BG (register 4). Points in up to 16 colors are displayed simultaneously. The background color/luminence is eet with the command: SETCOLOR 4, color, luminence. The color of the next foreground point to PLOT is set with command: COLOR (0-15).

THE ABOVE ARTICLE IS PROVIDED FOR YOUR INFORMATION ONLY. GTIA CHIP INSTALLATION BY ANY PARTY OTHER THAN AN AUTHORIZED SERVICE CENTER WILL VOID ANY WARRANTY REMAINING ON YOUR COMPUTER. IMPROPER INSTALLATION MAY CAUSE SERIOUS DAMAGE.

GTIA CHIP



DOWN MEMORY LANE

by Sheldon Leemon

This month, we will conclude our discussion of memory locations used for storage of data to be displayed on the screen. You will remember from last month that Atari computers are unusual, in that they do not have a fixed area of memory set aside for the screen display. Rather, the screen display area "floats" at the top of memory, changing location in accordance with changes in the graphics mode selected and memory size. In order to keep track of where this data area is at any given time, the computer keeps a couple of sets of memory locations as "pointers"; they contain the address of the start of the screen display area. One pointer tells the Operating System where this area is, and another pointer tells the display chip where to start the display. It is this second pointer that will be discussed here.

One of the neat things that this arrangement allows you to do is to "move" the screen display area without actually moving any bytes of data. This illusion of motion works like this: if you change the pointer so that what was the second byte of display memory is now the first byte, the whole screen will seem to shift one character to the left. If you change the pointer so that what was the 40th byte is now the first, the whole screen will appear to scroll up one line. This technique is known as coarse scrolling (as distinguished from fine scrolling, which moves the display one pixel at a time, rather than a whole character). You might remember a program published here called Outer Space Attack, that used this technique to move some menacing invaders towards your outpost. The following will demonstrate the simple method used to achieve this effect:

```
10 DH=PEEK(560)+ PEEK(561) *256 +5: DL=DH-1
```

First we find the two-byte pointer to display memory. As you may remember from my column on the display list, this pointer is located at the 4th and 5th bytes of the display list.

```
20 FOR I=1 TO 40: POKE DL, PEEK(DL) +1: FOR J=1  TO  50:NEXT
J:NEXT I
```

Next we add one to the low byte value of the pointer and then POKE the new value back in. This makes the whole display appear to move to the left. The loop which uses the J index was added to slow down the motion.

```
30 FOR I=1 TO 40: POKE DL,  PEEK(DL)  -1:  FOR  J=1  TO  50:
NEXT J: NEXT I
```

This line repeats the motion, this time to the right.

continued

These three lines show the basic logic used in the main program loop of Outer Space Attack. The only real difference is that there, provision was made to carry or borrow from the high byte when the low byte value got above 255 or below 0 (see Line 170 of that program).

That concludes our short hop down the lane. Next month, a stirring exposition of what the Central I/O utility means to me.

HIGH SCORES

Missile Command:	1,593,950	James Hetzel
	1,004,165	Dirk Hose

ERIC AAMODT (5 yrs. old) scored 11,990 on PAC-MAN. STEPHEN AAMODT scored 415,200 with 5 lives remaining when his mother made him turn it off to eat supper.

KEEP THOSE SCORES COMMING IN!

POKES FROM THE PREZ

I know February is a short month but it felt more like a week to me! I can't believe the numbers we're attracting to the general membership meetings...looks like we can expect regular crowds of three hundred or more. Please use common courtesy in parking your vehicles in the Community Center parking lot. If there 's no room left when you get there resign yourself to a walk of a block or two and park on one of the nearby residential streets.

MACE's AMIS (Atari Message and Information System) BBS is up and running 24 hours a day now. Modem users can reach AMIS at (313)-868-2064 for the latest in Atari news, notes, and downloadable programs. You'll find the first part of a series of articles on using the system in the April issue. Not mentioned in Sheldon's February meeting minutes, but still on our minds, is a MACE thank you to Spectrum Computers for providing any special cables and power strips we needed for AMIS, and to MACE member Jim Steinbrecher for the nice programs he has contributed for downloading.

The number of calls AMIS gets from WAY out of town is suprising. One of our newest MACE members is Lon Jones of Wheatland, Wyoming. Lon heard about us by calling into the Bulletin Board System and since its a little slow at the Two Bar Ranch this winter, he decided to join. MACE has become the first Atari Computer User Group in the world to top 300 in membership!

The quality of production and content of the MACE newsletter (or should I say magazine?) accounts for a great deal of our popularity. Kudos to newsletter Editor Marshall Dubin , Production Manager Richard Gizinski, and Advertising Manager and MACE VP Jerry Aamodt for their continuing terrific work! As our membership, operating budget and costs continue to grow your officers will remain committed (they lock us up sometimes to keep the members safe) to bringing you more and better services at reasonable cost.

Technical Editor Craig Chamberlain obviously knocked himself out to bring you the most complete rundown on the new GTIA chip published to date. I certainly felt completely run down after I finished reading all ten and a half pages of it! I know I'm going to have to read it several times to let it all sink in. Not sure if you have a GTIA in your machine? Type in this two line program and RUN it:

```
10 GRAPHICS 11
20 GOTO 20
```

If the screen turns black you have a GTIA, or your power just blew.

A note to beginners: Don't be afraid to read articles that seem to be over your head. Chances are that you will be able to pick up some of the concepts presented and you can build upon those. When I first started reading about microcomputers it all seemed like an alien language to me. But as you see the same terms used over and over again its possible to pick up a lot of what's going on just by the context of their use. And of course, when all else fails you can always read the relevant manuals. Remember, the difference between being a novice and an expert is the willingness to read material over until it starts to make sense.

continued

The MACE Birthday Party is set for the May general membership meeting, and attendance will be limited to MACE members only, given the turnout we are expecting. Earl Rice and Mark Cator from Atari will be here to visit, along with other surprises for your amusement. DON'T FORGET TO BRING YOUR MACE CARD! New folks will be able to purchase memberships at the door.

Chet Gonterman is now in charge of new cassette and disk submissions to the program library. Just bring your program contribution to the meeting and Chet will copy it on the spot. All programs submitted should be written by you or submitted with the permission of the author. We do not accept programs keyed in from Softside, but ones from Compute and Creative Computing are welcome (we have permission from those mags to use them).

We are now using a duplicating service for the cassette program library. This should solve some of the quality problems we were running into before, but since Tape Librarian Rod Graham is no longer a slave to the god CSAVE, our production costs have gone up. Cassette volumes are now the same price as the disks; four dollars each. All MACE media is guaranteed for one month by the club. You may return any failing media to the appropriate librarian for another copy of THE SAME volume.

Have you seen the March issue of Softside? MACE'S binary barrister, Sheldon "Instedit" Leemon just about owns the entire Atari section of the mag. Winning an ATARI Star Award has had some minor effect on Shel's professional demeanor. At his last court appearance, when asked by the judge to make his opening statement, Sheldon responded with "DIM A$(255)! GOSUB 3000!". continued

BUSINESS AND HOME COMPUTERS

SPECTRUM

THE COMPUTER STORE

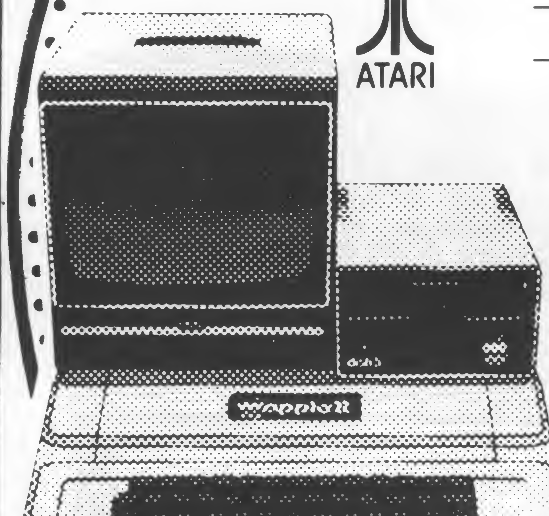
WE OFFER:

- A sales staff that gives personal service—and will answer all your questions.
- Up-to-date selection of business and personal software.
- A service department that cares for your equipment—in store or on site.
- Complete line of computers, accessories, supplies featuring:
  —Altos Hard Disk/multi-user
  —Apple Computer Systems
  —Atari Computers

ALTOS COMPUTER SYSTEMS

apple computer

ATARI



HOURS:
M-F 10 - 7:30 p.m.
Sat. 10 - 6:00 p.m.

SPECTRUM COMPUTERS

26618 Southfield Rd.
Lathrup Village, MI 48076
(Between 10 & 11 Mile)

Phone: 552-9092

MasterCard

VISA

AMERICAN EXPRESS

New software on the way! Automated Simulation's D&D game, TEMPLE OF APSHAI is finally ready to be released. Bill Budge's great pinball machine simulation, RASTER BLASTER will be out within a couple of weeks. Expect a very nice personal finance system from Synapse Software, the distributors of the well respected FILE-MANAGER 800 package. If you're interested in ATARI PASCAL, an advanced music synthesizer, or a Rubiks Cube solver, give APX a call. I believe all of these packages are orderable, but not officially announced yet.

Expect some major hardware announcements from Atari at the June NCC and CES shows. Looks like at least half of the officers plan on going to the Consumer Electronics Show in Chicago...we'll be making a full presentation of Atari and other show highlights at the June or July meeting.

Our next issue will include a MACE April Fool section as well as MICRO WARS, a look at the rivalries (real and imagined) between owners of different microcomputers. Should be interesting...oh well, I promised Marshall this would only be one page and I've blown that promise already. GAMBITS will be back next month. See you then.

Alan

Source ID: TCT987
Micronet: 70675,463

GTIA (continued)

The OS screen handler always uses a standard width playfield. The advantage of the narrow playfield is that less DMA is required, so programs execute faster. Unfortunately, the screen handler routines do not work properly when the playfield width is other than the standard. The wide playfield generates more data than the television can display; its uses are rather limited. It's even possible to turn off playfield completely, in which case ANTIC fills the screen with scan lines of the background color. As will be shown in a moment, the playfield also requires a "display list" so bit five must be set for any playfield type to be generated.

Remember that a byte is made up of eight binary "bits". If playfield and display list DMA is enabled, bits may be read from the computer memory during the course of one scan line. The bit pattern determines the frequency and intensity changes of the scanning beam, with the result being different color/luminances. The same bit pattern may be repeated for several scan lines. And, the bit pattern can be interpreted in different ways. This leads us to yet another definition.

A mode line is a contiguous group of scan lines for which display memory is read only once.

There are two main types of mode lines. In direct memory map modes, the bit pattern produces the same image on each scan line. Text modes are a more complicated mode type which use a character set. A discussion of text modes is beyond the scope of this article.

The ANTIC knows how to handle fourteen different kinds of mode lines. Each mode line corresponds to a different method for interpreting a bit pattern. A full screen graphics mode is actually just a series of identical mode lines. I smell a definition heading our way.

The display list is merely a sequence of bytes in memory that, among other things, tells ANTIC the proper sequence of mode lines for one screen.

Whenever the screen is opened (accomplished in ATARI BASIC with the GRAPHICS statement), the screen handler establishes a display list of many mode lines to produce a screen of the desired mode. Modes can be mixed by manually changing the display list. Display lists produced by the screen handler always contain the proper number of mode lines for exactly 192 scan lines of playfield. Altering the display list can affect the total number of scan lines, which is how the vertical size of the playfield is controlled.

The display list also has other functions, such as control of fine scrolling, horizontal blank interrupts, and loading the memory scan counter of the ANTIC so it knows where to start reading memory. For a more detailed discussion of the display list, the reader is referred to any of the several articles already written on that topic.

A mode line divided into several parts forms pixels, which are single plotting points somewhere within the playfield area. A pixel's vertical resolution is the same as the mode line in which it is displayed, so there can be just as many pixels vertically as mode lines in the display list. The number of color clocks over which one pixel

continued

ATTENTION COMPARISON SHOPPERS

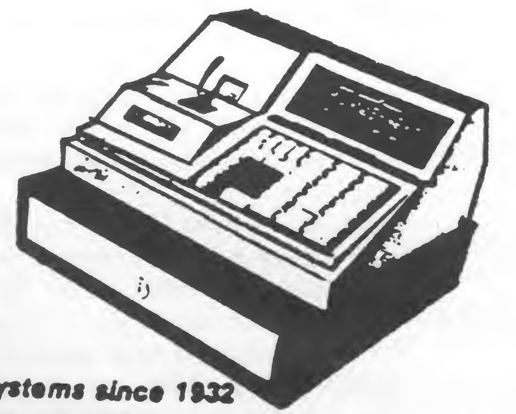
RWE

RITE WAY ENTERPRISES

RWE

EPSON MX-70
$299.00

THE
DISCOUNT
Computer
SOURCE



rc allen
cash register systems since 1932

MODEL #	# OF DEPARTMENTS	LIST	OUR PRICE
2001	9 TOTAL	$395	~~$249~~ 229
111A	6 DEPT.	$795	$495
2004	4 DEPT.	$895	$495
3004	4 DEPT.	$995	$595
3008	8 DEPT.	$1095	$695
3108/12	2 TO 12 DEPT.	$1195	$795
3016/48	16 TO 48 DEPT.	$1395	$895
9014/42	14 TO 42 DEPT.	$1295	$795
SUMIT SCALE		$790	$690

PAY LESS.

ATARI

GET MORE.

ATARI 800 with 16K memory	~~$749~~ 699
ATARI 400 with 16K memory	$349
ATARI 810 Disk Drive	$449
ATARI 850 INTERFACE	$169
software	15% TO 20% OFF
EPSON MX-80	$479
EPSON MX-100	$759

METRO DETROITS LARGEST
ATARI SOFTWARE SELECTION
15% TO 20% OFF LIST PRICE
GAMES*BUSINESS*PERSONAL*UTILITY
HASE SMART MODEM $239.
PRINTER PAPER
1000 8½ x 11 $10.95
3000 8½ x 11 $27.95
MOSAIC 32K BOARD $129.00
400/48K $579.00

HIGH PERFORMANCE FLOPPY DISK $1.99

751-2454

8262 12 MILE

WARREN, MI. 48093

GTIA (continued)

is spread is also determined by the mode line. Here is a little chart to show you the pixel size for the primary mapping modes.

MODE	COLOR CLOCKS	SCAN LINES	RESOLUTION (full/split screens)
3	4	8	40 by 24/20
4,5	2	4	80 by 48/40
6,7	1	2	160 by 96/80

Note that each time the width of a pixel is reduced, its height also decreases, so a single pixel appears to be square in shape regardless of the graphics mode.

Now to talk about memory. In the one color modes, one pixel is represented in memory by one bit. If the bit is on, playfield zero shows. If the bit is off, the background shows. Modes 4 and 6 are the one color modes. For more color, modes 3, 5 and 7 allow three colors. The tradeoff is that a single bit is no longer sufficient. Two bits, a pair, are required. The total value of the two bits selects either one of three playfields or the background.

BIT PATTERN	COLOR	PLAYFIELD TYPE
00	0	background
01	1	playfield zero
10	2	playfield one
11	3	playfield two

Playfield zero is the same thing as COLOR 1 in ATARI BASIC. Playfield one is really COLOR 2, and so on, with COLOR 0 being the background.

Although modes 4 and 5 both have the same resolution, or pixel size on the screen, mode 5 will require twice as much memory. In the lower resolution modes which require little memory in the first place, the additional memory needed is rather insignificant. You might have noticed that mode 3 had no single color counterpart. Consider that in a 48K system it is possible to have about 150 different mode 3 screens in memory simultaneosly. The chip designers probably decided it wasn't worth the effort (and memory savings) to provide a one color mode with such low resolution.

Therefore, the size of a pixel on the screen is determined by two things: how many scan lines high, and how many color clocks wide. The amount of memory required for a mode is also determined by two things: how many separate pixels to one mode line, and how many color possibilities per pixel. The only real connection between pixel size on the screen and size in memory is that bigger pixels fill up a screen faster, so there are less of them, and less memory is needed.

Okay, three colors means two bits must be used. Does that mean we are always stuck with only three colors which can't be changed? No. The CTIA is capable of generating 128 color/luminance variations. It can produce sixteen different colors, each in eight varying degrees of luminance. But 128 possibilities means seven bits would be required, which in most cases is simply not feasible. There is a limit as to how

continued

GTIA (continued)

much memory can be devoted to a screen. The solution to this problem is a sort of compromise, but it also presents some powerful and flexible advantages, too. The solution is to use color indirection.

With color indirection, the number of different playfields is limited according to the number of bits per pixel, but the actual color/luminance of each playfield can be one of the 128 possibilities. The data bits are used as an index or offset into playfield color registers. These playfield color registers use seven bits to select the color and luminance, as follows.

COLOR0	$02C4	708	playfield zero color register
COLOR1	$02C5	709	playfield one
COLOR2	$02C6	710	playfield two (used in modes 0 and 8)
COLOR3	$02C7	711	playfield three (used in color text modes)
COLOR4	$02C8	712	background color register

D7,D6,D5,D4	color
D3,D2,D1	luminance
D0	not used

BITS	VALUE	COLOR
0000	0	gray (no color)
0001	1	light orange
0010	2	orange
0011	3	red orange
0100	4	pink
0101	5	purple
0110	6	purple blue
0111	7	blue
1000	8	blue
1001	9	light blue
1010	10	turquoise
1011	11	blue green
1100	12	green
1101	13	yellow green
1110	14	orange green
1111	15	light orange

ATARI BASIC allows you to select a playfield color to draw in by using the COLOR statement. The color register that corresponds to that playfield can be changed by using SETCOLOR.

Color indirection is a power that should not be overlooked. It is possible to draw a detailed figure on the screen with one playfield, and then change the color of the entire figure with just one command. For example, a printed message can flash in colors to attract attention. A "glowing" effect can be created by rapidly changing the luminance of a playfield while maintaining the same color. Or, the playfield colors can all be set to the same color/luminance as the background. Figures drawn will not appear until the playfield color registers are changed. By changing the registers one at a time, an animation effect can be created. Color indirection may still not solve the problem of having many colors on the screen at the same time, but it does afford many possibilities that otherwise would be difficult to achieve.

continued

GTIA (continued)

In special instances, playfield color registers can be changed during the horizontal blank, but this requires machine code. Experience has shown that for many applications, three playfield colors can be sufficient.

Nevertheless, there are some cases in which many colors would be nice. Multiple colors is where the GTIA steps in. It should now be apparent that sixteen colors will require four bits per pixel, which is exactly the case. This is very expensive in terms of memory, so either pixel size will have to increase or display memory will have to increase. Because ANTIC has a limit as to how much memory it can access during one horizontal scan line, we have a limit as to how much memory can be devoted to a screen. Therefore, resolution will have to suffer.

Before I show you what the memory limit is, I want to talk about two modes which are exceptions to the above rules. Three things distinguish modes 0 and 8 from the normal modes. Each pixel is a half color clock wide; a side effect of this is artifacting. The background color now becomes the border, and the main part of the screen is filled with playfield two. Finally, since the whole screen is now playfield two, the bit no longer tells which playfield to use but which luminance to use.

MODE	BIT	LUMINANCE REGISTER
0,8	1	playfield one
0,8	0	playfield two (no image)

The color part of playfield one is ignored; only the luminance data is used. If the luminance values of playfields one and two are the same, the writing disappears. Modes 0 and 8 use this special half color clock, one playfield color, two brightness arrangement. Both modes have 320 distinct points of light horizontally and have single scan line resolution. The only difference between mode 0 and mode 8 is that the first is a text mode and the second is a direct mapping mode. Mode 0 uses a character set and thereby saves memory; about 1K is required for this mode. Mode 8 doesn't use a character set, and requires approximately 8K. That is our display memory limit. The ATARI 400/800 is not capable of doing DMA to much more memory than that during one television frame.

As I just showed, the "half color clock, one color, two brightness" mode is used by graphics modes 0 and 8. All the GTIA really does is provide three variations on this mode. They all use the maximum memory arrangement used by mode 8, so each of the three new modes requires 8K. All of the new modes use four bit pixels, so the horizontal resolution goes from 320 (half color clock) to 80 (two color clocks, as in modes 4 and 5). Therefore, the resolution for all three new modes is 80 by 192, for a total of 15360 points. One side effect of changing only the horizontal resolution is that the pixels are no longer square.

The ANTIC instruction register mode number for the maximum memory mode (the number you will find in the display list) is $0F, or decimal 15. It is important to understand that this number indicates not only mode 8, but 9, 10 and 11 as well. In fact, the display list for any one of these modes is identical to the display list for any of the others.

continued

GTIA (continued)

How then does ANTIC know which of the four is the desired mode? The answer is that ANTIC neither knows nor cares; no matter which mode is being used. ANTIC still has to do the same work of fetching memory. It's the GTIA that processes the video signal; somehow the chip must be told which of the four modes is wanted. The GTIA hardware register PRIOR does exactly that.

```
GPRIOR  $026F    623  (shadow)
PRIOR   $D01B  53275  (hardware)
```

The two most significant bits (bits 7 and 6) of this register are the GTIA special mode select bits. Here's how they are set.

MODE	BITS	HEX	DECIMAL
8	00	00	0
9	01	40	64
10	10	80	128
11	11	C0	192

For example, it is possible to switch from any one of the four modes to another simply by changing the values of the two select bits.

Other bits in GPRIOR serve different functions, so care must be taken not to alter them. These other bits allow multi-color players (blending on overlap), set all missiles to the color of playfield three to form a fifth player, and establish player/missile and playfield priorities. See the HARDWARE MANUAL for further information.

Now that we know how the three new modes are similar, let's find out how they are different.

Mode 11 is the one luminance, sixteen color mode. The overall luminance is set by the background color. It is now easy to draw rather finely detailed shapes in several colors without having to mess around with the display list and machine code interrupt routines. The thing I am especially excited about is going to make APPLE owners drool. The APPLE has a sixteen color mode with resolution of 40 by 48, called the "lo res" mode. Not only does the ATARI now have a sixteen color mode, but the resolution is eight times better than the APPLE's!

Sixteen colors does present a problem, though, since the GTIA has only four playfield color registers. Therefore, mode 11 does not allow color indirection. The color on the screen is determined directly by the bit data stored in memory; the values in the four color/luminance registers are ignored. Some may consider this a disadvantage, but there is a good point too. Just as the playfield color registers are not used, neither are the player/missile color registers, so by using players it is possible to have twenty one colors on the screen at the same time, without using display list interrupts or other tricks!

Mode 9 is the one color, sixteen luminance mode. This mode will be used to create some excellent three dimensional effects and digitized pictures. The sixteen luminances, when stacked vertically by the scan line with each line having the next brightest luminance, blend so well it is very difficult to see the division from one to the other. The main color is set by the background color. Weird things happen when

continued

GTIA (continued)

you change the luminance of the background. Another nice fact is that having sixteen main colors with sixteen luminance variations means that the ATARI is capable of producing 256 colors!

One advanced application for mode 9 is the display of digitized pictures. Digitization is a process by which a normal television picture, such as from a station or video recorder, can be analyzed and divided into different luminances. That information can be stored on disk for later display. Mode 9, with sixteen luminances and rather high resolution, is able to reproduce such pictures with impressive quality. Thus far we have seen only for digitized pictures. They were apparently made by some people at ATARI, and two of the pictures were, well, uh, for mature viewers only. Standing from a short distance, however, it is very difficult to tell whether any of these pictures is computer generated or not. I have never seen such quality on any other computer in the 400/800 price range without expensive additional equipment.

Mode 10 is a cross between the other two modes; it allows 8 colors plus the background, each with its own luminance as in the primary modes. Unlike the other two modes, this one allows color indirection, so it uses the playfield and player/missile registers for color/luminance information. This chart shows how data values correspond with playfield registers.

BITS	VALUE	REGISTER	PLAYFIELD
0000	0	704	PCOLR0
0001	1	705	PCOLR1
0010	2	706	PCOLR2
0011	3	707	PCOLR3
0100	4	708	COLOR0
0101	5	709	COLOR1
0110	6	710	COLOR2
0111	7	711	COLOR3
1000	8	712	COLOR4
1001	9	712	COLOR4
1010	10	712	COLOR4
1011	11	712	COLOR4
1100	12	708	COLOR0
1101	13	709	COLOR1
1110	14	710	COLOR2
1111	15	711	COLOR3

Only nine of the sixteen possible data values correspond to different playfields. Data values greater than eight just repeat playfields. For some reason, the background color is no longer set by COLOR4, but instead by PCOLR0. The ATARI BASIC statement SETCOLOR can't be used to change the player/missile color registers, so the equivalent POKE must be used. For any register, the data part of the POKE is the color choice number muliplied by sixteen, plus the luminance (refer to earlier chart).

The power of indirection is magnified when eight main drawing colors can be used. This mode is very useful for creating motion effects. With nine color/luminances and color indirection, mode 10 may prove to be the most versatile of the three new modes.

continued

GTIA (continued)

Remember that the GTIA only controls how the display is generated, so all programs written for the CTIA will run on a GTIA machine in the same way. There can be no such thing as incompatibility. We have, however, come across one discrepancy between the CTIA and GTIA. The video signal generated by the GTIA is shifted one half color clock, so colors produced by artifacting, such as in POOL 1.5 or Jawbreakers, will be different. That is just a minor visual difference; the important thing is that all software will be entirely compatible. Of course, you cannot expect a CTIA to generate these three new modes, but again the conflict is the display, not the program. In fact, I don't think it is even possible for the computer to tell whether it has a CTIA or GTIA in it.

Because of the half color clock shift, it is now possible for players and playfields to overlap perfectly, whereas before with the CTIA they didn't.

There have been some cases where software will not run on GTIA machines. This is due to the fact that some of the new computers with the GTIA also have a revised (no bugs) operating system in them. ATARI has made very clear which memory locations and vectors are permanent and protected from any revisions. If a program does not run on a GTIA machine, it is totally the software producer's fault because illegal entry points were used.

One other conflict has appeared which really surprised me. It turns out that a few programs written on CTIA machines carelessly set the GTIA special mode select bits of GPRIOR. Since these two bits do nothing on the CTIA, there was no problem. But there was also no reason to mess with them! When the same programs are run on a GTIA, the accidental bit settings affect the display, even though modes 9,10, and 11 are not used. I have been told that Crush, Crumble, and Chomp is one such program. The function of those two bits has not been a secret; I figured out their function several months ago by reading the source listing. If the authors in question did not know the functions of those two bits, in my opinion they should not have messed around with something they knew nothing about at the expense of consumers. I personally feel that any program which exhibits this problem is an example of inept programming.

There is a difference between the normal modes and the three new modes - the three new ones do not allow split screen (text window at bottom) configurations. If you remember how modes 8 and 0 are related, you should understand why. The mode used in the text windows is mode zero, which follows the special "half color clock, one color, two luminances" arrangement. Having the mode select bits in GPRIOR set for a mode greater than 8 causes mode 0 to act funny. A split screen would only be possible if a display list interrupt was inserted just before the text window area. The interrupt routine would have to reset to zero the mode select bits in the hardware register PRIOR, not the shadow register. The hardware register will then be reset to the value of GPRIOR during the vertical blank service routine.

The three new modes do seem to handle player/missile to playfield collisions a little differently. In modes 0 and 8, a playfield two

continued

R. Gyzinski
435-6026

MICHIGAN ATARI COMPUTER ENTHUSIASTS

President: Arlan Levitan
12709 Borgman
Huntington Woods, MI 48070
399-6963
Micronet: 70675,463
Source ID: TCT987

Vice Pres.: Jerry Aamodt
4148 Huhn
Rochester, MI 48063
574-1020

Treasurer: Judy Braun
39281 Eliot
Mt. Clemens, MI 48043

Secretary: Sheldon Leemon
14400 Elm
Oak Park, MI 48237
398-2608

Programs: Gary Luzier
22530 Gordon
St. Clair Shores, MI 48081
773-3446

Librarians: Sam Findley (Disk)
8171 Denwood, Apt. #1
Sterling Heights, MI 48077
939-3822

Rodney Graham (Tape)
12270 Deming
Sterling Heights, MI 48077
264-6355

Newsletter: Marshall Dubin
2639 Hempstead
Auburn Heights, MI 48057
338-3488

PROGRAM LIBRARY SUBMISSION FORM

I wish to submit the following program to the M.A.C.E. program library. I warrant that I have good title to this program, and that it does not infringe upon any copyright. Limited rights are transferred herewith for the use of this program within the membership of M.A.C.E. and its associated Atari Computer organizations.

Program Name ______________________ Submission Media: Tape ____ Disk ____ List ____
Description ____
Catagory: Business ____ Demonstration ____ Education ____ Game ____ Utility ____ Other ____
Minimum System: Memory ____ K BASIC ____ Assembler ____ PILOT ____ PASCAL ____
Disk ____ Tape ____ Interface ____ Modem ____ Printer ____ Light Pen ____
Joystick Controllers ____ Paddle Controllers ____ Keyboard Cont. ____
Name ______________________ Phone __________ Date __________

M.A.C.E.
P.O Box 2785
Southfield, MI 48037

$15.00 fee for 12 months*

M.A.C.E. MEMBERSHIP APPLICATION

Name ______________________ Phone __________
Street ____
City __________ State __________ Zip Code __________
Company (if applicable) ______________________
System Description ______________________ Disk/Tape __________
Suggestions ____
I can help with . . . ____

☐ New
☐ Renewal

*Make checks payable to:
M.A.C.E.

Date __________ Coupon __________
Amount __________ Membership Card Number __________
☐ Cash Expires __________
☐ Check Number

GTIA (continued)

collision is flagged when a player or missile hits a pixel whose luminance is controlled by COLOR1 rather than the COLOR2 for the main playfield. From what I have been able to tell thus far, there is no kind of playfield collision at all in modes 9 and 11. Mode 10 collisions work only for playfield colors that correspond to the usual playfield registers (COLOR1 through COLOR3). Also, the fact that the background in this mode is set by PCOLR0 affects the priority of players and playfields in some cases.

The GTIA still allows only 8 luminances on the normal modes.

All new ATARI computers are being shipped with the GTIA at no extra cost. The CTIA is no longer being produced. The new machines with the GTIA have little yellow or white stickers that have the letter "G" on them. Those of us who have older machines with the CTIA can replace it with a GTIA at a nominal cost. The part number is C014805. Twenty five dollars seems to be the going rate for the chip plus installation.

It will be a simple matter to replace the CTIA. The CTIA is on the CPU card that plugs into the motherboard inside the ATARI case. It's not soldered in so the replacement operation should take only thirty minutes if you have taken your computer apart before. For the meantime, if you don't have the GTIA, don't fret. It will be a while before much software requiring the chip is available.

Well, there you have the full story of the GTIA. The best personal computer for under $1000 has just gotten better. It's just further proof that ATARI is the way to go.

OF SPECIAL INTEREST

The following computer stores have offered to provide space at their stores for meetings of MACE special interest (SIG) groups:

BYTE BY BYTE- 52070 VAN DYKE, UTICA, 254-2698

COMPUTER HORIZENS- 37099 SIX MILE ROAD, LIVONIA, 464-6502

RAINBOW COMPUTERS- 16 MILE AND ROCHESTER ROAD, TROY

If you wish to use these facilities, contact the store managers for more information. Space is on an availablity basis.

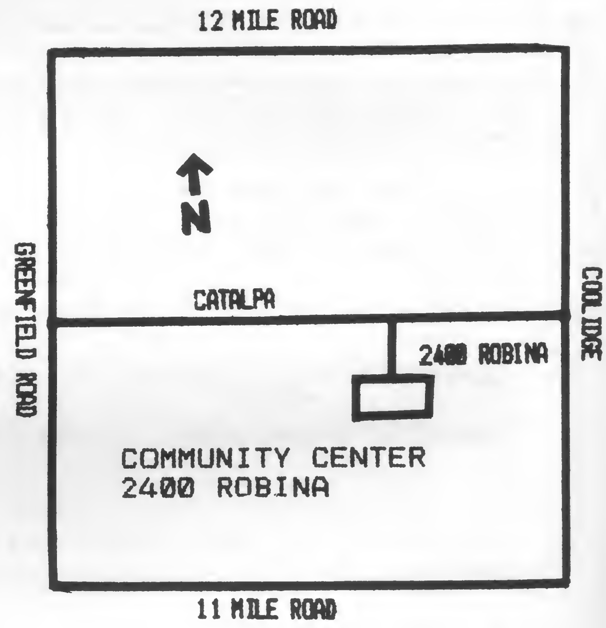
A BIG M.A.C.E. THANK YOU! to these stores for their cooperation!

M.A.C.E.

1982 MEETING SCHEDULE

MARCH 18	SEPTEMBER 16
APRIL 15	OCTOBER 21
MAY 20	NOVEMBER 18
JUNE 17	DECEMBER 16

BERKLEY COMMUNITY CENTER
2400 ROBINA
BERKLEY, MICHIGAN



UNCLASSIFIED ADS

8K MEMORY, $30.00
Guarenteed operational
Rick Montgomery, 781-6639

MICHIGAN ATARI COMPUTER ENTHUSIASTS
P.O. BOX 2785
SOUTHFIELD, MICHIGAN 48037

MAR 1982 PM 1982

01/83

IMPORTANT DATED MATERIAL

PLEASE DO NOT DELAY

Printing and Bindery Services by • GRAPHIC ENTERPRISES, INC. • Detroit, Michigan • 313-839-6800